Vorlesung Betriebssystemarchitektur WS 2003/04

Aufgabenblatt 5 vom 8. Januar 2004

(Vorstellung der Lösungen bei den Tutoren bis zum 30. Januar 2004)

Aufgabe 5.1: (40 Punkte)
Schreiben Sie in der Programmiersprache C ein Programm, welches eine dateibasierende Datenbank implementiert. Konkret geht es um eine Geburtstagsdatenbank. Es gibt eine Tabelle mit zwei Spalten "Name" und "Geburtstag". Ihr Programm soll zwei Funktionen unterstützen: Lesen und Schreiben. Beim Lesen soll zu einem angegebenen Namen der Geburtstag ausgeben werden. Die Schreib-Funktion soll unter dem angegebenen Namen den Geburtstag speichern. Beispiel:

Schreiben:
```
$ aufg51  db.dat  Finja  20.12.2002
$
```

Lesen:
```
$ aufg51  db.dat  Finja
Name: Finja  Geburtstag: 20.12.2002
$
```

Die Daten sollen in einer Datei (im Beispiel: db.dat) gespeichert werden. Die Datei ist in 48-Byte große Datenbank-Einträge aufgeteilt. Die Einträge haben folgenden Aufbau:

```
struct db_entry
{
    char name[32];
    char geburtstag[16];
};
```

Der Name und der Geburtstag sollen als 0-terminierte Zeichenkette gespeichert werden. Die maximale Länge des Names beträgt 31 Zeichen, die des Geburtstages 15 Zeichen. Die Position der Datenbank-Einträge in der Datei wird durch eine vorgegebene Hash-Funktion bestimmt
(`http://www.dcl.hpi.uni-potsdam.de/uebung/blatt5/hash.c`).

Schreibvorgang: Es wird zunächst der Hash-Wert x des Namens bestimmt. Dann wird der momentane Dateizeiger auf den x-ten Datenbank-Eintrag in der Datei positioniert (Byte-Position x*48). Nun wird durch eine Leseoperation festgestellt, ob an dieser Position schon ein Eintrag vorhanden ist. Wenn nicht (alle Bytes sind 0 oder Dateiende wurde überschritten), dann wird der Eintrag an dieser Stelle gespeichert. Wenn bereits ein Eintrag vorhanden ist, wird überprüft, ob der Name identisch ist. In diesem Fall wird an dieser Stelle der neue Geburtstags-Eintrag gespeichert. Wenn der Name nicht identisch ist, dann wird der Dateizeiger auf den nachfolgenden Datenbank-Eintrag gesetzt und obige Prozedur wiederholt (Lineare Sondierung bei Hash-Kollision).

Lesevorgang: Es wird der Hash-Wert x des gesuchten Namens bestimmt. Dann wird der momentane Dateizeiger auf den x-ten Datenbank-Eintrag in der Datei positioniert. Nun wird durch eine Leseoperation festgestellt, ob an dieser Position ein Eintrag vorhanden ist. Wenn nicht, dann gibt es zu diesem Namen keinen Datenbankeintrag. Wenn ein Eintrag vorhanden ist, muss zunächst der Name des Eintrags mit dem gesuchten Namen verglichen werden. Sind beide gleich, ist der gesuchte Eintrag gefunden und kann ausgegeben werden. Wenn der Name nicht identisch ist, dann wird der Dateizeiger auf den nachfolgenden Datenbank-Eintrag gesetzt und obige Prozedur wiederholt.

Sie können das Programm unter Linux oder Windows entwickeln. Es sollen die Funktionen open(2), read(2), write(2) und lseek(2) benutzt werden (auch unter Windows). Verpacken Sie alle Source-Code-Dateien inklusive eines Makefiles in eine ZIP-Datei und schicken Sie diese ZIP-Datei als Attachment mindestens **48 Stunden** vor dem Tutoriumstermin an `bs@hpi.uni-potsdam.de` mit dem Betreff
`AUFGABE=5.1 GRUPPE=`<Ihre Übungsgruppennummer> `OS=WIN32` oder
`AUFGABE=5.1 GRUPPE=`<Ihre Übungsgruppennummer> `OS=LINUX` je nach Betriebssystem.
Die ausführbaren Dateien sollen `aufg51.exe` (Windows) bzw. `aufg51` (Linux) heißen.

Erklären Sie Ihrem Tutor das Programm und führen Sie es vor.
Wie groß wird die Datenbank-Datei, wenn Sie einen Geburtstag unter dem Namen "Michael" speichern? Wieviel Platz wird dafür auf der Festplatte wirklich verbraucht? (Linux: du(1), Windows: Datei komprimieren)

Aufgabe 5.2: (40 Punkte)
Schreiben Sie ein C-Programm, welches die Dateinamen des momentanen Verzeichnisses ausgibt. Der Inhalt von eventuell vorhandenen Unterverzeichnissen soll rekursiv ausgegeben werden. Die Ausgabe der speziellen Verzeichnisse "." und ".." soll unterdrückt werden. Beispiel:

```
C:\aufg52> aufg52.exe
aufg52.c
aufg52.exe
Makefile
test
test\tst.txt
C:\aufg52>
```

Sie können das Programm unter Linux oder Windows entwickeln. Lesen Sie sich dazu die Dokumentation zu den Bibliotheksfunktionen opendir(3), readdir(3), stat(2) durch bzw. suchen Sie in der MSDN-Bibliothek nach FindFirstFile und FindNextFile. Verpacken Sie alle Source-Code-Dateien inklusive eines Makefiles in eine ZIP-Datei und schicken Sie diese ZIP-Datei als Attachment mindestens **48 Stunden** vor dem Tutoriumstermin an `bs@hpi.uni-potsdam.de` mit dem Betreff
`AUFGABE=5.2 GRUPPE=`<Ihre Übungsgruppennummer> `OS=WIN32` oder
`AUFGABE=5.2 GRUPPE=`<Ihre Übungsgruppennummer> `OS=LINUX` je nach Betriebssystem.
Die ausführbaren Dateien sollen `aufg52.exe` (Windows) bzw. `aufg52` (Linux) heißen.

Erklären Sie Ihrem Tutor das Programm und führen Sie es vor.